AMENDMENTS TO THE CLAIMS:

This listing of claims will replace all prior versions, and listings, of claims in the application:

LISTING OF CLAIMS:

1.(original) Method for generating application software for managing a process, characterized by the fact that said method implements a system software that is common to all of the application software and that said method includes:

- a process representation stage (32) that implements a very small number of classes of actions or generic objects, typically less than 20, in at least one diagram of the application;

- a stage (33) for transcribing each object of each diagram into an action corresponding to an attributed object, whereby during the transcription stage each class of action or generic object is associated with an application data capture interface;

- a stage (34) for transcribing the nodes, branches, and leaves of each diagram into an action corresponding to an attributed object;

- a precompilation stage (35) during which it is verified that the object attributes required for the operating logic of the

application are present and are supplied appropriately in terms of syntax;

- a compilation stage (36) during which the data descriptors of the attributed object are integrated and are assembled with the system software to produce an executable application software; and

- a stage (37) for execution of the executable application.

2.(original) Method according to Claim 1, wherein, during the object transcription stage (33), at least one action launches a complete processing cycle that is located at a remote location of a tree structure that corresponds to at least one diagram and, once said processing cycle is completed, returns to its starting point.

3.(currently amended) Method according to claim 1 ~~one of Claims 1 or 2~~, wherein, during the execution stage (37) the executable application software implements a library for managing the sequence of events corresponding to the above-mentioned at least one diagram, whereby said library constitutes an automaton (70) that manages the sequence of events of the processes and executes the operations that checkpoint them, whereby the sequences of events of the operations are defined in the

application referential, by means of the method (10), by describing the actual data flows.

4.(currently amended) Method according to claim 1 ~~one of Claims 1-3~~, wherein, during the compilation stage (36) or the execution stage (37), the method employs an engine (71) that includes an executive that is responsible for recognizing the hardware and communication configuration.

5.(original) Method according to Claim 4, wherein the engine (71) manages one or more databases according to a data file descriptor that is provided by the application referential, that is, a list of the information contained in each file and the list of the access indices, whereby a list of the fields constitutes each of these indices, the links between the multiple encodings of a single item in multiple services, multiple sites, or multiple companies.

6.(original) Method according to Claim 5, wherein the databases are synchronized according to a schedule that is determined by the diagram, upon demand, or before certain predefined events.

7.(currently amended) Method according to claim 1 ~~one of Claims 1-6~~, wherein the object transcription stage (33) includes, for programming each action:

i) a naming stage (331) during which a name is given to said action;

ii) a function definition stage (332), during which said action is identified with a task; and

iii) an information definition stage (333) during which the information that will be processed in the action is identified.

8.(original) Method according to Claim 7, wherein the compilation stage (36) replaces the action name that is given by the transcriber during the naming stage (331) with an index in a task table.

9.(currently amended) Method according to claim 1 ~~one of Claims 1-8~~, wherein, during the representation stage (31) and transcription stage (32, 33), said at least one diagram corresponds to at least one tree structure in which the nodes and leaves, where the code is implemented, are made up of actions, whereby the return values of these actions determine the movement in the tree structure.